DevOps

DevOps

— это методология разработки, которая помогает наладить эффективное взаимодействие разработчиков с другими IT-специалистами. Это набор процессов и инструментов, которые позволяют компании создавать и улучшать продукты быстрее, чем при использовании традиционных подходов к разработке программного обеспечения.

— это комбинация слов «разработка» (**dev**elopment) и «эксплуатация» (**op**eration**s**), которая отражает процесс интеграции этих дисциплин в единый непрерывный процесс. Разработчики и тестировщики отвечают за Development, а администраторы — за Operations.

Модель методологии напоминает вертикальный поток, где все фазы последовательно перетекают одна в другую. После этапов определения требований к продукту и его проектирования наступают этапы написания и сборки кода. Затем код отправляется в руки тестировщиков. После проверок отдел эксплуатации загружает код на рабочие машины — продукт запущен.

Знание DevOps необходимо разработчикам для более глубокого понимания их продуктов и оптимизации расходов на запуск кода, а также позволяет ускорять запуск приложений. Методология позволяет гибко приспосабливаться к меняющимся условиями и снижать стоимость разработки и обслуживания. Применяя DevOps, разработчики могут оперативнее замечать и исправлять ошибки и реагировать на запросы заказчиков.

Одна из практик DevOps — **непрерывная интеграция и доставка (Continuous integration и Continuous delivery или CI/CD)** — это методы, которые автоматизируют процесс выпуска программного обеспечения, начиная от сборки и вплоть до развертывания. CI/CD помогает свести к минимуму ошибки, повысить темпы сборки и качество разрабатываемого продукта за счет автоматизации.

Что DevOps дает команде

- **Меньше ошибок.** Одна из причин, почему случаются сбои при развертывании, связана с багами. В DevOps циклы разработки короче обычных, поэтому код выходит чаще. В результате искать ошибки становится проще, а значит, количество сбоев уменьшается.
- **Сокращение времени выхода сервиса на рынок.** Масштабируемые инфраструктуры — облачные платформы, инструменты для ускорения сборки, параллельные рабочие процессы, работа в одной среде — сильно сокращают время работы. Развертывать и запускать приложение стало в разы быстрее.
- **Создание более гибких и отказоустойчивых систем.** Это достигается за счет использования облачной инфраструктуры. Она дает возможность быстро масштабировать систему, использовать только нужное количество ресурсов и оперативно увеличивать мощности.
- **Повышенная надежность и безопасность приложений.** Среди DevOps-инструментов есть те, которые анализируют исходный код программного обеспечения, чтобы определить, есть ли в нем недостатки безопасности. Еще есть приложение, которое сканирование сервисы на наличие в них уязвимостей — OWASP (Open Web Application Security Project).

Роль тестировщика в процессе непрерывной поставки

Главный принцип **DevOps** это непрерывное тестирование (*continuous testing, CT*). Оно должно начинаться еще на этапе согласования требований и завершаться на этапе мониторинга “прода”.

Непрерывное тестирование — то, что работает в пределах CI/CD вообще. Прежде чем организовать непрерывное тестирование как процесс, нам нужно сформировать стратегию тестирования: выделить группы тестов, которые мы будем запускать в разное время, чтобы проверить, насколько корректно работает приложение.

В CI-фазе важно, чтобы сборка проходила быстро, поэтому чаще всего применяют облегчённые типы тестов:

- unit-тесты — чтобы протестировать отдельные компоненты;
- интеграционные тесты — чтобы проверить интеграцию этих компонентов;

- базовый линтинг — чтобы проверить код на соответствие тому, как мы договорились писать его внутри команды;
- статистический код-анализ — чтобы отловить потенциальные уязвимости в нашем коде;
- smoke-тесты — чтобы проверить, развернулось ли приложение, и корректно ли работают его базовые компоненты.

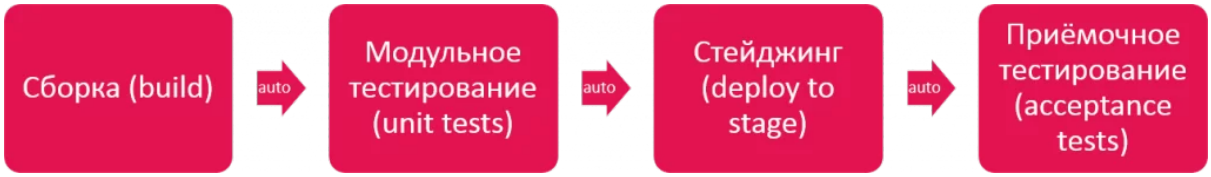
Можно делать больше тестов, но обычно этих 5 при условии, что они хорошо написаны, достаточно. Запустив их, мы уже предупредим около 90% багов на стадии CI.

QA-команда играет важнейшую роль в DevOps, начиная раннюю автоматизацию и помогая находить дефекты как можно раньше, создает тестовые наборы для этапов DevOps для инсталяционного и Smoke-тестирования.

Этапы непрерывной поставки

Непрерывная интеграция (CI)

— первичный, базовый процесс обновления ПО, в рамках которого все изменения на уровне кода вносятся в единый центральный репозиторий. Такое внесение принято называть слиянием. После каждого слияния (которое проходит по несколько раз в день) в изменяемой системе происходит автоматическая сборка (часто приложение упаковывается в Docker) и тестирование (проверка конкретных модулей кода, UI, производительности, надёжности API). Таким образом разработчики страхуются от слишком поздних обнаружений проблем в обновлениях.

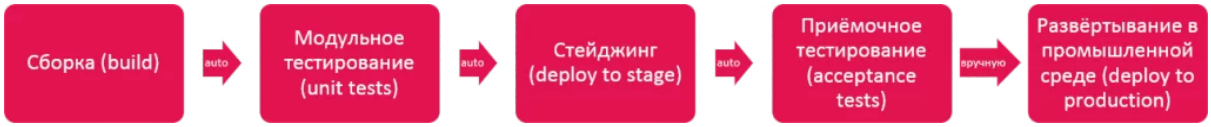


Непрерывная интеграция. Каждый этап запускается и выполняется автоматически

Непрерывная доставка (CD)

— CI + CD. Следующий после CI уровень. Теперь новая версия не только создаётся и тестируется при каждом изменении кода, регистрируемом в репозитории, но и может быть оперативно запущена по одному нажатию кнопки

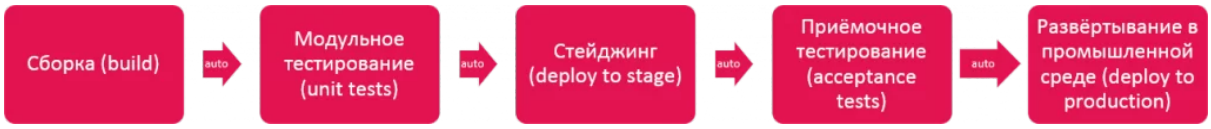
развёртывания. Однако запуск развёртывания всё ещё происходит вручную — ту самую кнопку всё же надо кому-то нажать. Этот метод позволяет выпускать изменения небольшими партиями, которые легко изменить или устранить в случае необходимости.



Непрерывная доставка. Развёртывание выполняется автоматически, но запускается вручную

Непрерывное развёртывание (CD)

— CI +CD + CD. После автоматизации релиза остаётся один ручной этап: одобрение и запуск развёртывания в продакшен (злосчастная кнопка). Практика непрерывного развёртывания упраздняет и это, не требуя непосредственного утверждения со стороны разработчика. Все изменения развёртываются автоматически.



Непрерывное развёртывание. Все этапы запускаются и выполняются автоматически

Quality Gates

— это заранее определенные этапы, во время которых проект проверяется на соответствие необходимым критериям для перехода к следующему этапу. Quality Gates являются важным компонентом официальных процессов управления проектами, используемых различными организациями.

Цель Quality Gates – обеспечить следование набору определенных правил и передовых практик, чтобы предотвратить риски и увеличить шансы на успех проекта. С помощью качественных Quality Gates организации могут гарантировать, что руководители проектов выполняют свою работу и не пропускают никаких важных шагов.

Процесс реализации

Quality Gates основаны на чек-листах, по которым менеджеры проектов должны пройти на разных этапах жизненного цикла проекта. Эти чек-листы включают в себя ряд вопросов, касающихся различных аспектов проекта, включая объем работ, бюджет, заинтересованные стороны, риски и соответствие требованиям.

Перед совещанием по вопросам качества руководитель проекта изучает соответствующий чек-лист по определенным Quality Gates и отвечает на каждый вопрос, принимая во внимание текущий статус проекта. Он также предоставляет заполненный чек-лист соответствующим лицам, принимающими решения, чтобы дать им достаточно времени для изучения информации до совещания по качеству.

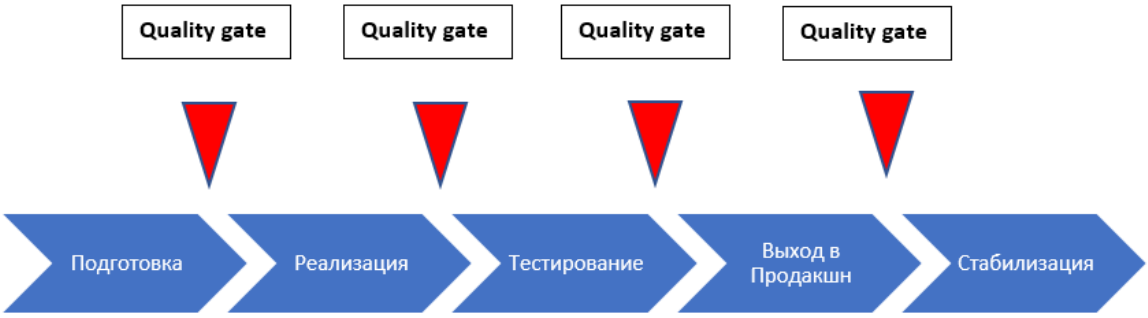
Во время совещания участники знакомятся с чек-листом и обсуждают наиболее важные пункты чек-листа. Менеджер проекта предоставляет контекст и отвечает на любые возникающие вопросы.

Обсуждение в основном вращается вокруг вопросов, которые не были завершены – пунктов чек-листа, на которые были даны ответы “Нет” или “В процессе”. Возможно, какая-то роль в проекте еще не заполнена или бюджет еще не подписан клиентом.

В зависимости от темы и серьезности проблемы, команда Quality Gates, состоящая в основном из руководства, спонсора проекта и ключевых заинтересованных сторон, может начать предпринимать различные действия. Если у кого-то из команды на данный момент нет работы, это не проблема, и проект может продолжаться. Однако, если бюджет проекта все еще обсуждается, проект следует приостановить до тех пор, пока не будут утверждены расходы.

Человек, который контролирует соблюдение Quality Gates, может также потребовать принятия дополнительных мер для конкретных пунктов чек-листа. Если, например, одна из заинтересованных сторон выражает озабоченность по поводу того, соответствует ли планирование ресурсов правилам управления персоналом, она может отправить запрос, чтобы отдел кадров проверил план ресурсов проекта.

В своей практической реализации Quality Gates организованы в виде совещаний, которые запланированы в конце каждого этапа проекта. Вот как это обычно выглядит:



Рассмотрим Quality Gates для каждой из фаз разработки продукта (SDLC).

Фаза SDLC	**Необходимые практики**	**Quality gates**
Фаза исследования продукта	Тренинг для разработчиков	Команда разработчиков прошла тренинг Tech Lead/Dev Lead есть на проекте
Анализ требований	Анализ рисков	Определен Product Data Owner Проведена классификация данных Определены требования и их соответствие Проведен анализ рисков по приложению
Разработка дизайна	Произведена оценка архитектуры приложения Произведено моделирование архитектуры Созданы макеты приложения	Созданы документы по архитектуре приложения Создана модель возможных угроз
Разработка продукта	Статический анализ кода Code Review	Произведен анализ найденных дефектов Найдены угрозы и риски, разработаны шаги по нивелированию их воздействия Все дефекты должны быть устранены либо должны быть приняты компенсационные меры
Тестирование продукта	Acceptance Testing Динамический анализ кода Функциональное тестирование	Все устраненные дефекты проверены Произведен анализ причин возникновения дефектов с приоритетом Medium и выше
Развертывание продукта	Оценка конфигурации деплоймента	Формальная приемка продукта в процессе финального анализа продукта

	Финальный анализ продукта	